

LKC
QA
76.9
.N38
F475
1989



Gouvernement du Canada
Ministère des Communications

Government of Canada
Department of Communications

Le Centre canadien de recherche sur l'informatisation du travail
Canadian Workplace Automation Research Centre

2.1 **SUN-TULIP USER GUIDE**

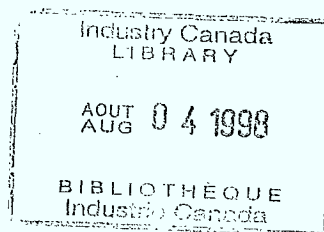
by

1/ **Ron Ferguson** /

QA
76.9
N38
F475
1989
0-3

Canada

QA
76.9
N38
F475&
1989



2. / **SUN-TULIP USER GUIDE** /

by

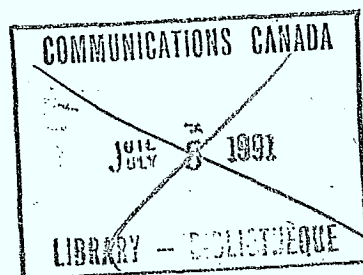
/ **Ron Ferguson** /

This report is the result of research done at the
Canadian Workplace Automation Research Center
under the direction of Joël Muzard, Expert Systems Group.

Department of Communications

Laval

November 1989



Nº de cat. Co 28-1/41-1989E
ISBN 0-662-17472-0

The opinions expressed are those of the author only.

* Une version française de ce document est aussi disponible.

QA
76.9
N38
F4452
1989
C.3

DD 9382576
DL 10640972

Table of contents

<u>Introduction</u>	3
<u>Levels of Comprehension</u>	4
<u>Conceptual Comprehension</u>	4
<u>Syntactic Comprehension</u>	5
<u>Word Matching</u>	5
<u>Limitations of Tulip's Conceptual Comprehension</u>	5
<u>Entering and Exiting Tulip</u>	6
<u>Implementation of the help function of Tulip system</u>	7
<u>Aborting Tulip's Execution</u>	7
<u>Input to Tulip</u>	8
<u>Commands</u>	10
<u>The Form of English Assertions Accepted</u>	11
<u>The Form of English Queries Accepted</u>	11
<u>Answering Queries by Word Matching</u>	12
<u>Formulation of the statements</u>	
<u>Conjunction</u>	13
<u>Noun Phrases</u>	13
<u>Determiners</u>	14
<u>Pronouns</u>	16
<u>Verbs</u>	16
<u>Names</u>	17
<u>Dates and Times</u>	18
<u>Arithmetic Operations</u>	18
<u>Error Messages</u>	18
<u>French Text</u>	18
<u>Expressing Knowledge in Tulip</u>	19

Introduction

TULIP (Text Understanding and Logical Inferencing Processor) is a natural language interface to an 'electronic scratchpad'. The user can make assertions and ask questions of the system in a subset of English. The user can also enter rules describing relationships between entities.

In addition to knowledge which the user supplies, Tulip has built-in expertise in certain areas. Specifically, Tulip is able to interpret business cards, dates and times, and certain arithmetic operations.

This user's manual describes the operation of Tulip on Sun Workstations. It is assumed that the user has a basic familiarity with Sun Workstations.

In this user's manual, examples are given of input which Tulip does and does NOT accept. By an abuse of a standard convention in linguistics, input which is unacceptable to Tulip will be preceded by [*].

[*] Green ideas sleep furiously.

Levels of Comprehension

Tulip can be said to 'understand' English, in that it is able to use assertions and rules to find responses to queries. However the user should not expect Tulip to understand English to the extent that a human understands it. In order for a computer program to understand a natural language (such as English) at the human level, the program would have to have the experience, knowledge, and inferencing capabilities that a human has. No currently existing program has these capabilities.

In order to degrade gracefully, Tulip has three levels of comprehension. If Tulip is unable to understand a statement at the highest level of comprehension, it tries the next level down. The levels of comprehension from highest to lowest are :

- [1] Conceptual comprehension.
- [2] Syntactic comprehension.
- [3] Word matching.

Conceptual Comprehension

At this level of comprehension, Tulip is able to relate different ways of saying the same thing, even when different words are used. For instance, Tulip assumes that the following assertions mean the same thing :

Mr. John Smith works for Kentek.
Kentek employs Mr. John Smith.

Thus Tulip is aware of a relationship between the words 'works for' and 'employs'.

Syntactic Comprehension

At this level of comprehension, Tulip is able to parse a sentence and is able to relate sentences which say the same thing using different syntactic forms. For instance, the following sentences give Tulip the same information :

Kentek employs Mr. John Smith.
Mr. John Smith is employed by Kentek.

In the above example, the second sentence is the passive form of the first sentence. Tulip also understands different ways of expressing the possessive :

the father of Jack
Jack's father

Tulip can also relate queries to assertions if the same expressions are used in both :

Kentek employs Mr. John Smith.
Who is Mr. John Smith employed by?

Word Matching

If a question is input which Tulip is unable to parse, Tulip finds those previously entered assertion(s) which contain the most words in common with the question. These assertions are displayed as possible answers to the query. Often this simple technique is able to retrieve the information the user wants, although irrelevant information may also be retrieved.

Limitations of Tulip's Conceptual Comprehension

Tulip has limited built-in conceptual comprehension. It understands most user input at the syntactic level rather than the conceptual level. Therefore, the user should attempt to employ a consistent terminology when using Tulip. The user can extend Tulip's conceptual comprehension by entering rules which define the relationships between words.

Entering and Exiting Tulip

To use Tulip you must be in the following directory :

/home/condor2/tulip

Load Tulip by typing :

tulip.bin

Tulip is a large program so it will take some time for the program to load. Once Tulip has loaded the following Quintus Prolog prompt will appear :

| ?-

Enter Tulip by typing :

tulip.

When Tulip is ready for user input, the following prompt appears :

Enter a sentence :

To exit from Tulip type :

exit.

(or)

quit.

This will return you to the Quintus Prolog prompt :

| ?-

To exit from Quintus Prolog type :

halt.

Implementation of the help function of Tulip system

To implement the help function while your in Tulip, type :

help.

Aborting Tulip's Execution

The user can abort Tulip while it is executing by typing ^C (control-C).
The following prompt is displayed :

Prolog interruption (h for help)?

To abort from Tulip type :

a

This will put one in Quintus Prolog. One can re-enter Tulip by typing :

tulip :

Input to Tulip

The input to Tulip consists of the following basic types :

- [1] Commands - A command is a line of input which begins with a keyword (eg. read, save, restart) and ends with a period followed by a carriage return. They are used to tell Tulip to perform special functions. An example of a command is :

restart.

- [2] Assertions - An assertion is a line of english input which makes a statement of fact and ends with a period followed by a carriage return. Tulip remembers assertions and uses them to answer queries. An example of an assertion is :

Bill is the husband of Mary.

If the first word in a sentence is not a proper name, it is not necessary to capitalize it.

a cat is on the mat.

- [3] Queries - A query is a line of english input in the form of a question which ends with a question mark followed by a carriage return. When the user enters a query, Tulip tries to find an answer to the query based on what it has been told. An example of a query is :

who is Mary's husband?

It is not necessary to capitalize the first word of a query.

- [4] Rules - A rule is a line of English-like input of the form :

<Conclusion> if <Premise1> (and <Premise2>...).

where <Conclusion> and the <Premise>s are simple statements which may contain 'variables' which act as standin's for objects. A variable is a capitalized letter other than 'A' or 'I'. Rules are used to establish relationships between concepts. An example of a rule is :

X is the wife of Y if Y is the husband of X.

Using this rule and the assertion given above that :

Bill is the husband of Mary.

Tulip can answer the question :

Who is Bill's wife?

The meaning of the verbs 'have' and 'is' depends on the type of object the verb takes. As a result, a variable can NOT be used as the object of the verbs 'have' or 'is' :

[*] X has Y if X owns Y.

[*] X is a Y if the type of X is Y.

However, a variable can be used as the subject of the verbs 'have' or 'is' :

X has a window office if X is a manager.

X is a mammal if X has fur.

Often, a rule which involves just one variable can also be expressed as an assertion. For example the preceding rules can be expressed as :

Every manager has a window office.

Everything that has fur is mammal.

- [5] Business Cards - If a line is entered which does not end with a period or question mark, Tulip assumes that it is the first line of a business card. If the first line of a business card happens to end with a period, a space should be typed after the period and before the carriage return. Tulip displays the following prompt to indicate that it expects more lines of input for the business card :

I :

A blank line terminates the business card input.

Commands

save <name> .

- saves the sentences the user has typed in since the last Save or Restart command. The sentences are saved as <name>, where <name> is a alphanumeric word. If <name> has previously been used in a save command, the sentences are added to the end of <name>. Sentences which are saved can be read back in using the Read command.

read <name> .

- reads in the sentences which were saved using 'Save <name>.' Note : The 'save <name>.' command saves sentences in a file called '<name>.text' in Tulip's 'knowledge' directory. It is possible to put a text file in the knowledge directory and read it using the 'read' command. Also, the knowledge files can be edited using an editor.

restart.

- Erases Tulip's memory

debug.

- displays Tulip's processing of a sentence. This is very detailed and will probably not be useful to the average user.

end debug.

- turns off the detailed display of the processing of sentences.

help.

- explains how to use Tulip.

The Form of English Assertions Accepted

In general, Tulip accepts English statements which express a literal statement of fact.

Negative information can not be expressed. Thus one can not enter a statement with the word 'not' in it. eg.

[*] George is not a programmer.

Statements with the word 'or' are also not accepted. eg.

[*] Helen or Mary is married to Jim.

An assertion must begin with a noun phrase which is the subject of the statement.

The Form of English Queries Accepted

Tulip accepts both Wh and yes/no questions. A Wh question is a question which begins with a question word such as 'who', 'what', 'which', 'where'. 'How many' is also considered to be a Wh word. The Wh word can be preceded by a preposition. The following are examples of Wh questions :

Who is Frank's wife?

To whom is Frank married?

How many people work for Kentek?

Which employees live in Laval?

Note that the wh word must be the first word (or second word if it is the object of a preposition) in the sentence. Thus the following query is not allowed :

[*] Jack is married to whom?

The above query should be expressed as :

To whom is Jack married?

or

Whom is Jack married to?

The word 'how' by itself is not handled.

Jack runs quickly.

[*] How does Jack run?

A *yes/no* question is a question which can be answered by 'yes' or 'no'. Such questions usually begin with the verb 'is' or 'does' :

Does Frank work for Kentek?

Is Frank a manager?

Answering Queries by Word Matching

If a question is input which Tulip is unable to parse, Tulip uses word matching to try to find relevant information. Thus if one enters keyword(s) followed by a question mark, Tulip will display those assertions which contain the largest number of keywords.

Conjunction

Only certain uses of the word 'and' are allowed. One can use 'and' between statements :

Joe lives in Toronto and Henry lives in Montreal.

If the subject is the same in both statements, it can be omitted in the second statements :

Joe lives in Guelph and works in Toronto.

Relative clauses (see noun Phrases) can also be joined by 'and'.

Noun Phrases

A noun phrase can be either a common noun phrase, eg.

the tall man

or a proper name, eg.

John Doe

The words in a proper name must be capitalized.

A common noun phrase (but not a proper noun phrase) can be modified by a (conjunction of) relative clause(s). For instance, the following is a noun phrase.

the man who lives in Halifax and is employed by Acme Inc.

Tulip does not handle reduced relatives, such as :

[*] the man employed by Acme Inc.

This should be written as :

the man who is employed by Acme Inc.

Common noun phrases can be either singular or plural :

the cat
the cats

However, conjunctions of noun phrases are not allowed. For example,

[*] the men and women
[*] Bill and Jane

are not allowed.

A list of words inside double quotes is treated as a singular proper noun.

"The Young and the Restless" is a television program.

Determiners

A common noun phrase typically begins with a determiner such as 'a', 'the', 'some', 'all', or 'every'. The determiners 'most' and 'few' are not allowed.

If an assertion begins with the determiner 'a', eg :

a bear has claws.

Tulip can not determine whether the user means :

some bear has claws or every bear has claws.

Therefore, Tulip displays a menu with these two choices and asks the user to choose the appropriate meaning.

If 'a' is used in any position other than the beginning of a sentence or is used meaning 'some' at the beginning of a sentence, Tulip assumes that a new object is being referred to. In contrast, if 'the' is used, Tulip assumes that the user is referring back to an object that was mentioned in a previous sentence. Thus, if the statement :

George has a car.

has been entered, and the user then enters :

Helen likes the car.

Tulip assumes that 'the car' refers to the last mentioned car, ie. George's car.

On the other hand, if the user had entered :

Helen likes a car.

Tulip would not assume that it was George's car that Helen likes.

The only determiner allowed with a proper noun is 'the' :

The Volvo
[*] a Volvo.

Pronouns

Tulip will attempt to find the referent of third person singular pronouns such as 'he', 'she', 'him', 'her', and 'it'. It will also try to find a referent for possessive third person singular pronouns such as 'her' and 'his'.

Jack works for Acme Inc.
He owns a car.
Alice is his wife.
She gave him his coat.
who owns a car?
who is Jack's wife?
what did Alice give to Jack?

In the above example, 'he', in the second sentence, is assumed to refer to Jack, the last mentioned male.

Plural pronouns, such as 'they', 'them', and 'their', are NOT handled.

Verbs

Tulip accepts verbs in the present, past, or future tense. However, the tense of verbs is only used to interpret dates. Apart from that, the tense of a verb is ignored.

Modal verbs (verbs which modify other verbs) such as 'can', 'should', etc., are also accepted but are ignored.

Thus the following sentences are given the same meaning by Tulip :

Tweety can fly.
Tweety should fly.
Tweety could fly.
Tweety will fly.
Tweety flies.

Other than modal verbs, Tulip can not parse sentences which contain verbs that take other verbs as arguments (eg. persuade) :

[*] Tim persuaded Tom to hire Mary.

Names

Tulip is sometimes able to automatically determine what type of object a proper name refers to. Tulip knows common first names for men and women. Tulip uses this knowledge to try to class the people mentioned in a sentence as being men or women. Similarly, Tulip will assume that proper names which end with 'Limited', 'Incorporated', or 'Company', refer to organizations. Tulip also knows the names of cities, provinces, and states.

When a name appears in a sentence, Tulip tries to match the name with an object that it already knows about. The name does not have to exactly match the already established name of the object, but the new name must be a subset of the original name. For example, if Tulip knows about a person called 'Mr. Jack Smith' is being referred to. In the case where a name could refer to more than one person that Tulip knows about, the most recently mentioned person is chosen. Thus if 'Mr. Jack Smith Jones' has been mentioned more recently than 'Mr. Jack Jones'. When a person is first mentioned, his most complete name should be used since that individual can only be referred to by that name or subparts of that name from that point onwards.

Tulip does not allow an object to have more than one name.

Dates and Times

See the 'dates' demo for examples of Tulip's use of dates and times.

Arithmetic Operations

See the 'cars' demo for examples of Tulip's use of 'average', 'maximum', 'total', and 'minimum'.

Error Messages

If Tulip can not parse an assertion or query, it displays the following message :

Syntactic parser failed towards the end of this segment :

It then displays as much of the input sentence as it was able to parse. Usually the last word displayed in the sentence is the one that Tulip had trouble with.

If Tulip is able to parse a query but is not able to find a solution, Tulip asks the user if she wants a list of potentially relevant facts. If the user answers 'Yes', Tulip uses word matching to find those assertions which have the most words in common with the query.

French Text

Tulip has limited ability to handle French text. In general, Tulip is not able to parse French sentences. However Tulip can use word matching on certain French words. Its French vocabulary is limited to words which might appear on a business card and terms used in the field of artificial intelligence.

To use French accents, one must include the following line in one's .cshrc file :

```
setenv DEFAULT_FONT "/home/condor/critter/fonts/f_screen.r.14"  
setenv EMACS_FONT  "/home/condor/critter/fonts/f_serif.r.14"
```

and have the following lines in a .emacs_pro file in one's home directory :

```
(load "/home/condor/critter/.emacs_pro")  
(load " ../isabelle/emacs/dead_key_accents")
```

French accents can be added to letters by using the keys marked R1 to R5 which appear to the right of the main keys on a Sun 3. The accents are added by typing the appropriate R key followed by the letter the accent is to be added to. The accents associated with the R keys are :

- R1 - aigu
- R2 - grave
- R3 - cidille
- R4 - trima
- R5 - circonflexe

Expressing Knowledge in Tulip

Tulip can be used to express inheritance of properties between classes of objects. For instance, one can enter :

```
every bird can fly.  
every robin is a bird.  
Ernie is a robin.
```

If one then asks :

Can Ernie fly?

Tulip will respond 'Yes'. Since the user has stated that every robin is a bird, all robins will inherit the properties that are associated with birds. (eg. flying).

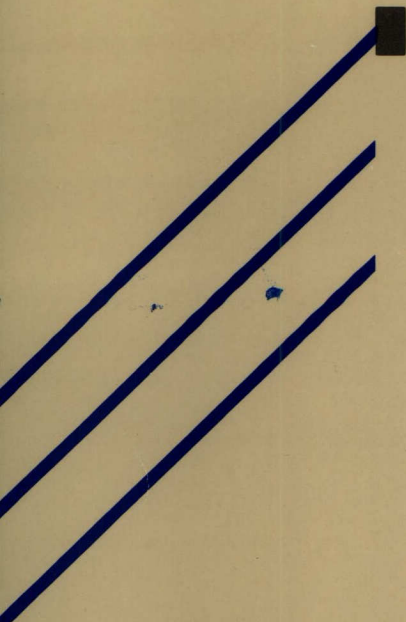


LKC
QA 76.9 .N38 F475 1989
Sun-Tulip user guide

DATE DUE

[illegible]

Pour plus de détails,
veuillez communiquer avec :



*Le Centre canadien de recherche
sur l'informatisation du travail*
1575, boulevard Chomedey
Laval (Québec)
H7V 2X2
(514) 682-3400

●
● ●
For more information,
please contact:

*Canadian Workplace
Automation Research Centre*
1575 Chomedey Blvd.
Laval, Quebec
H7V 2X2
(514) 682-3400